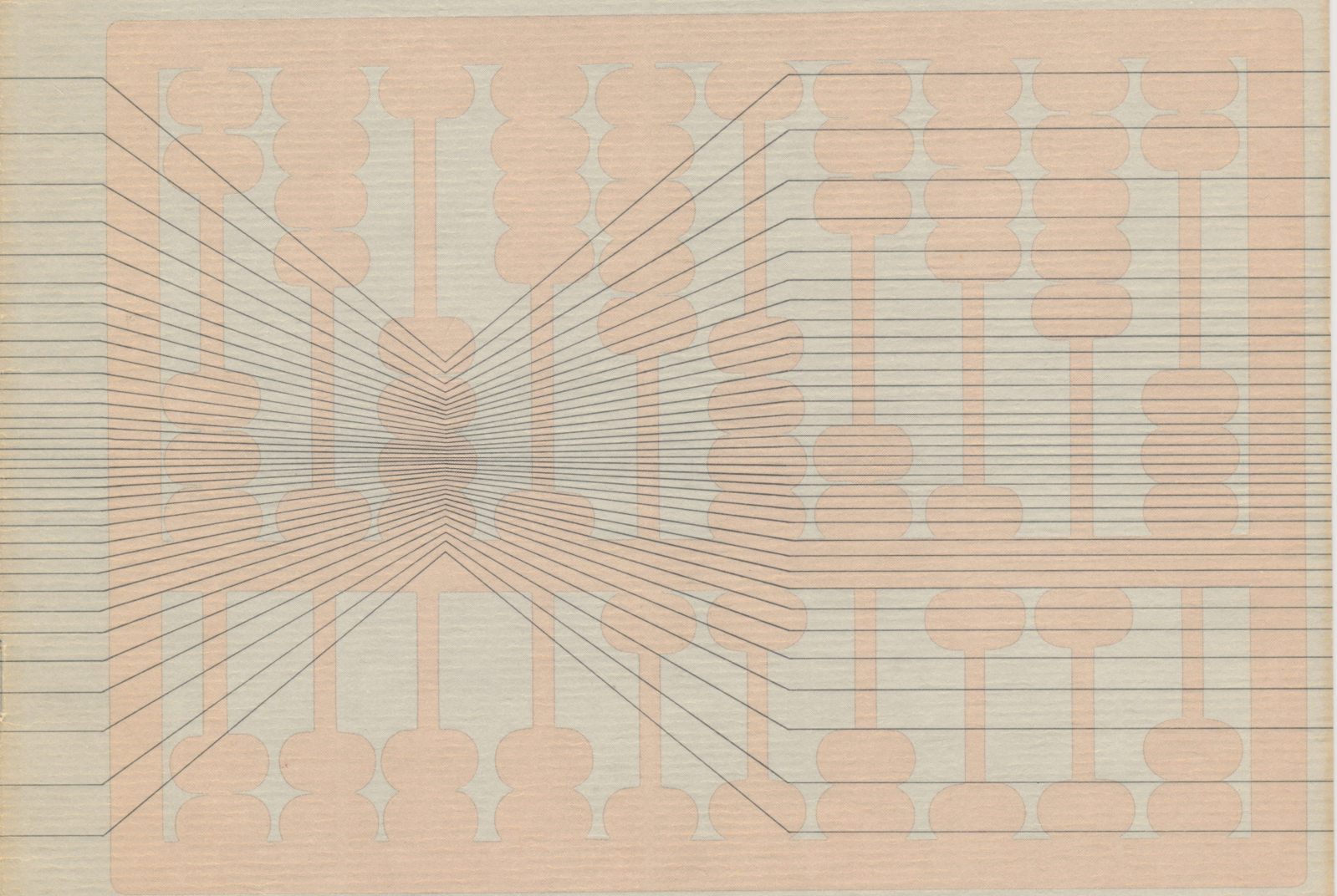
I. P. Sharp Associates

SHARP APL
Text Editor



SHARP APL TEXT EDITOR

Printed February 1978.

Printed in Canada.
Publication Code: J24FE01

CONTENTS

	Page
Introduction	1
Sample Letter	2
Entering Text	3
Editing and Printing	5
Making Modifications	6
The State Setting Functions	13
Using the SHARP APL File Subsystem in Editing	17
Appendix I: Summary of Edit Control Commands	24
Appendix II: Summary of Text Correcting Instructions	26
Appendix III: Summary of State Setting Functions	28
Appendix IV: Summary of File Handling Functions	29

INTRODUCTION

The I.P. Sharp Text Editor is a system for entering, correcting and formatting text. Text which has been entered into the system, can be retrieved and modified at any time, thus eliminating the task of retyping an entire document each time it is changed or updated.

The procedure used to prepare a document consists of two stages, each of which is associated with a specific function. The function *CH* is used to enter and modify text and the function *EDIT* is used to format and print text. In stage 1, the text for a document is entered along with "edit control commands" which determine the format of the output. This input (text plus edit control commands) is called "raw text". During stage 2, the text which has been entered is formatted according to the instructions given by the embedded edit control commands. These commands are then removed and a formatted copy of the document is produced.

The function *CH*, which is used to input raw text, is also used to modify it so that corrections can be made to the raw text as it is being entered. In addition, if the formatted output produced with *EDIT* appears to need further modification, it is possible to return to stage 1 and make any necessary changes with *CH*. A formatted version of the modified text is then produced by proceeding again to stage 2 and executing *EDIT*. The production of a typical document usually involves several passes through each of the two stages.

The Text Editor exploits the SHARP APL File Subsystem by permitting the "raw text" entered in stage 1, as well as the formatted text produced in stage 2, to be stored on file. In this way, large quantities of raw text can be stored and then accessed whenever modifications are necessary. In addition, multiple copies of the edited document can be produced without having to invoke the *EDIT* function each time.

The output of the Text Editor is page-oriented. Each page is generated separately. In recognition of the fact that the APL character set does not include lower case alphabetic characters, provision has been made for mixing two different typefaces in the output (usually APL and Courier). When printing a page containing two typefaces, on a terminal which supports only one typeface at a time, the page must pass through the terminal twice - first for the printing of the principal typeface, then for the printing of the second typeface. If the page contains only one typeface, only one pass is necessary. Printing on the highspeed printer requires specification of the *APLFULL* print train when both typefaces are present.

This manual will explain the conventions of the SHARP APL Text Editor by progressing through each of the stages involved in a typical editing cycle. The text for a sample letter will be entered into the system (stage 1) and edited (stage 2). Then, returning to stage 1, the raw text will be modified and subsequently re-edited (stage 2 again). This cycle will involve many of the commonly used features of the Text Editor. Sections on the State Setting Functions and the use of the Text Editor in conjunction with the SHARP APL File Subsystem appear in the second half of the manual. Complete lists of operators and commands can be found in the Appendices.

SAMPLE LETTER

Suppose the user wishes to use the Text Editor to produce the following letter. The lines have been numbered for reference purposes.

```
                                November 4, 1977.                    1
Acme Products Limited,                                               2
123 Main Street,                                                     3
Waxen Apple, Saskatchewan.                                           4
                                                                     5
Dear Gentlemen:                                                      6
     This is to introduce you to the Text Editor of the              7
SHARP APL system, available from I.P. Sharp Associates               8
Limited. The Text Editor is a program package designed to            9
help the user prepare edited output.                                 10
     If you should have enquiries, please contact I.P. Sharp         11
Associates Limited or your local SHARP APL representative.           12
                                                                     13
                                Yours sincerely,                     14
                                                                     15
                                                                     16
                                                                     17
                                Ralph Morrison.                      18
```

Before entering text into the system, it is helpful to have a general idea of the shape the final document is to take. In the case of the above letter, it is important to note the following:

1. Line 1 is either to be right justified or tabulated to begin at a certain point on the page.

2. Lines 2, 3 and 4 must each start at the left margin.

3. Line 5 is necessarily blank.

4. Line 6 necessarily starts on a new line at the left margin.

5. The paragraphs which begin on lines 7 and 11 must be indented.

6. Line 13 is necessarily blank.

7.. Line 14 should be lined up with line 1.

8. Lines 15, 16 and 17 are necessarily blank.

9. Line 18 should be aligned with lines 1 and 14.

Once the user has a general idea of the format the finished document is to take and has selected and installed the principal print element (in this case, Courier) he is ready to proceed to stage 1 and begin entering the text.

Note: Throughout this manual it is assumed that, except during the execution of functions such as *CH*, *EDIT* and *STATE*, the terminal has a Courier print element attached. It is also assumed that the Text Editor tabs are in their default state (i.e., tabs set every 5 spaces).

ENTERING TEXT

To begin entering the text for the sample letter, the user first accesses the *EDIT* workspace and executes the function *CH* by entering the following two commands:

```
)LOAD 4 EDIT
CH
```

A certain area of memory called *RINP* is set aside for the entry of raw text. When *CH* is first executed, it will respond with a zero indicating that *RINP* is currently empty. This is the signal to start entering text. Because any document is almost certain to be longer than one line, the user enters an appropriate amount of text and then hits carriage return (CR) at the end of a convenient word. When a CR is entered, *CH* prints the number of characters in *RINP* and waits for more text to be entered. When all the text has been entered, the user enters an empty input line (i.e., two consecutive CRs) to signal the termination of text entry. For example, the terminal session in which the raw text for the sample letter is entered would look as follows. (CRs have been added to indicate the position of each carriage return.)

```
      CH
0
!t7November 4, 1977.!sAcme Products Limited,!sl23CR
49
Main Street,!sWaxen Apple, Saskatchewan.!s!sDearCR
98
Gentlemen:!iThis is to introduce you to the TextCR
147
Editor of the SHARP APL system, available fromCR
194
I.P. Sharp Associates Limited. The Text Editor is aCR
246
program package designed to help the user prepareCR
296
edited output.!iIf you should have enquiries,CR
342
please contact I.P. Sharp Associates Limited or your localCR
401
SHARP APL representative.!s!s!t7Yours sincerely,!sCR
452
!s!s!s!t7Ralph Morrison.CR
476
CR
!t7November 4, 1977.!sAcm 25
CR
```

Notice that after each line of input *CH* responds with the total number of characters currently in *RINP*. To signal the termination of text entry, two consecutive CRs are entered - one after the last input line and one after the final "total number of characters" response. The double carriage return signals to *CH* that it should change from a data entry function to a data correction function. *CH* indicates that it has changed and is ready for corrections by producing a small window of text (the first 25 characters of input). Assuming that no modifications need to be made to the raw text at this point, the user signals to the Editor to exit entirely from *CH* (and stage 1) by hitting CR once more.

Several "edit control commands" have been used in the raw text shown above. These are as follows:

!t7 causes a tab to the seventh tab position (in this case, because the tabs are set at every 5 spaces, the characters following the command will be printed starting after position 35).

!s is the command to start a new line. It is used before each of the three lines of the address and twice before "Dear Gentlemen" to force a blank line.

!i is the command to start a new line at the indent position. !s!i will produce a blank line before the beginning of a paragraph.

These three edit control commands come from a much larger set which will be discussed in the sections of this manual which follow. Either upper or lower case letters can be used (i.e., !s and !S have the same effect).

Having made the assumption that no changes need to be made to the raw text at this time, the user is now ready to proceed to stage 2 and have the text edited and printed.

EDITING AND PRINTING

To have the raw text for the sample letter converted to the required format, the user enters

EDIT

At this point, the terminal keyboard unlocks to enable the user to:

- insert an appropriate piece of paper or to advance the paper to a new sheet

- type in a header line or page number if either is desired

On pages other than the first, entering the edit control indicator, " or !, causes the **previous** page to be reprinted.

When the user is ready, he hits CR and the page begins printing. The function *EDIT* is in control until the page is completely typed. When printing is complete, the print element will come to rest either at the left margin or in the centre of the page. If it is at the left margin, the page is complete and may be taken out. If there is a second (third and so on) page to be printed, the user inserts another sheet of paper. When the user hits CR, the second page will be printed.

If, after the printing of a page, the print element is in the centre of the page, it is necessary to roll back the paper to the starting position (usually one line above the first printed line) and install the appropriate print element. When the user hits CR, *EDIT* will print those parts of the page which are to be appear in the second typeface. The option which permits two typefaces to appear on the same page is governed by another of the edit control commands which will be introduced in the next part of this manual.

Only one typeface has been used in the sample letter so that once *EDIT* has been entered and a CR given after the pause, the letter which appears in the section entitled "Sample Letter" will be printed.

MAKING MODIFICATIONS

After inspecting the output produced in stage 2, the user may decide that modifications are in order. For example, suppose the sample letter is to be modified as follows:

```
                                    November 4, 1977.                    1
                                                                         2
Acme Products Limited,                                                   3
123 Main Street,                                                         4
Waxen Apple, Saskatchewan.                                               5
                                                                         6
Dear Gentlemen:                                                          7
     This is to introduce you to the Text Editor of the                  8
SHARP APL system, available from I.P. Sharp Associates                   9
Limited. It is a program package designed to help the user              10
prepare edited output.                                                  11
                                                                        12
     If you should have enquiries, please contact your local            13
SHARP APL representative or:                                            14
                                                                        15
               I.P. Sharp Associates Limited                            16
               145 King Street West                                     17
               Suite 1400                                               18
               Toronto, Canada                                          19
               M5H 1J8                                                  20
               (416) 364-5361                                           21
                                                                        22
     Thank you for your interest in SHARP APL.                          23
                                                                        24
                                    Yours sincerely,                    25
                                                                        26
                                                                        27
                                    Ralph Morrison.                     28
```

It is important to note the ways in which this letter differs from the first. "Text Editor" has been underlined and *SHARP APL* appears in the APL typeface. The second sentence of the first paragraph now begins with "It" instead of "The Text Editor". A blank line has been inserted between the first and second paragraphs. The second paragraph has been modified to contain the address block (lines 15-20). In addition, a third paragraph has been added to the letter.

Note that lines 15-20 (the address block) are to start neither at the left margin nor at the indented position. This can be accomplished by entering an appropriate tab value before each of these lines or by temporarily resetting the left margin. It is the second method that will be employed here in order to demonstrate another of the edit control commands.

To produce this second sample letter, it is necessary to return to stage 1 and change the contents of *RINP*. One way of doing this would be to first clear out the old contents of *RINP* by typing *CLEAR* and then enter the text and edit control commands necessary to produce the new letter. This would involve the following terminal session. (CRs have been added to indicate the position of a carriage return.)

```
      CLEAR
      CH
0
!t7November 4, 1977.!sAcme Products Limited,!sl23CR
49
Main Street,!sWaxen Apple, Saskatchewan.!s!sDearCR
98
Gentlemen:!iThis is to introduce you to theCR
142
!uText Editor!n of the !msharp apl!n system,CR
187
available from I.P. Sharp Associates Limited. ItCR
236
is a program package designed to help the userCR
283
prepare edited output.!s!iIf you should haveCR
328
enquiries, please contact your local !msharp apl!n CR
380
representative or:!hl5!s!sI.P. Sharp AssociatesCR
427
Limited!sl45 King Street West!sSuite 1400!sCR
471
Toronto, Canada!sM5H 1J8!s(416) 364-5361!h00!sCR
517
!iThank you for your interest in !msharp apl!n.!sCR
566
!s!t7Yours sincerely,!s!s!s!s!t7Ralph Morrison.CR
613
CR
!t7November 4, 1977.!sAcm 25
CR
```

Several new edit control commands have been introduced:

- **!u** starts underlining text which follows (in this case, Text Editor).
- **!m** indicates that the text which follows is to be printed in alternate typeface (in this case, using APL print element).
- **!n** stops both underlining and alternate typeface which were requested by !u and !m.
- **!h15** creates hanging indent 15 spaces in from the old left margin. Any appropriate number could have been used instead of 15. Note that the combination !h15!s is used since !h affects the **following** line of edited output.
- **!h00** cancels the hanging indent. The left margin returns to normal.

Note: It is not possible to use !u to underline text which is to appear in the alternate typeface since !u cancels !m and vice versa. Text which is to appear in the APL typeface can be underlined (letters only) by backspacing and underlining in the raw text.

The contents of *RINP* will now produce the modified letter. However, there is a more efficient method of altering the contents of *RINP*. Retyping an entire document, especially in the case of a large document, is needlessly time-consuming. As was stated earlier, the function *CH* can be used to correct raw text as well as to enter it. Rather than retyping the entire letter, the raw text which is already in *RINP* can be modified by employing *CH* as a text correcting function. To demonstrate how this can be done, assume that *RINP* still contains the raw text originally entered for the first sample letter. Entering *CH* will produce the following result:

```
      CH
!t7November 4, 1977.!sAcm 25
```

Because *RINP* already contains raw text, the Editor assumes that *CH* is entered as a data correcting function rather than a data input function. It responds with a "window" showing the first twenty-five characters in *RINP*. This window can be moved forward by using an appropriate text correcting instruction until the user reaches the portion of raw text which is to be modified. He then spaces over to the characters in the window which are to be modified and makes any necessary changes using another text correcting instruction. Upon hitting CR, *CH* will verify that the change has been made by printing the revised window. For example, the first change to be made to *RINP* is the insertion of !u before "Text Editor". The terminal session which follows shows how this is done. Note that the numbers on the left have been added for reference purposes.

```
         CH
1. !t7November 4, 1977.!sAcm 25
2. lText
3. Text Editor of the SHARP  168
4. .!u
5.  the !uText Editor of the 163
```

When *CH* is executed, it displays the first 25 characters in *RINP* (line 1) and waits for some text correcting instruction to be entered below this window. The first modification to be made is the underlining of the words "Text Editor". To move to the appropriate window, the user enters (in line 2) l (for locate) and the word he wishes to locate. *CH* searches through *RINP* until it locates the first occurrence of the word "Text" and then displays the appropriate window in line 3. In line 4, the user requests that !u be inserted **before** the letter "T" using the appropriate text correcting instruction (. for insert). In line 5, *CH* verifies that the change has been made by printing the modified window. Note that each time a window is displayed *CH* indicates its position in *RINP* by printing a number to the left. For example, the number 163 after line 5 indicates that the last character of this window is the 163rd character in *RINP*.

There are several other changes which need to be made to *RINP* before it can be used to produce the modified letter. The terminal session in which all the necessary modifications are made, appears on the following pages. To make it easier to follow, the session has been divided into units with an explanation of each unit appearing below it. In addition, each line of the terminal session has been numbered on the left. Note that throughout the session, odd-numbered lines indicate a response of the function *CH*, while even-numbered lines correspond to user commands or input.

```
        CH
1.  !t7November 4, 1977.!sAcm 25
2.  lText
3.  Text Editor of the SHARP  168
4.  .!u
5.   the !uText Editor of the 163
6.                    .!n
7.  ditor!n of the SHARP APL  176
8.                ////////!msharp apl!n
9.   the !msharp apl!n system 186
10. lThe
```

Lines 1 through 5, in which !u is inserted before "Text Editor", were explained above. In line 6, the user spaces over and inserts !n before the blank which follows "Editor". This will stop the underline. *CH* verifies the insertion in line 7. In line 8, the user deletes SHARP APL and replaces it with !msharp apl!n to obtain the alternate typeface. *CH* verifies this replacement in line 9. In line 10, the user asks *CH* to locate the word "The".

```
11. The Text Editor is a prog 259
12. dtor
13. ted.  is a program packag 254
14.      .It
15. ted. It is a program pack 254
16. loutput
17. output.!iIf you should ha 324
18.         .!s
```

CH locates the next occurrence of "The" and displays it in line 11. In line 12, the user deletes the characters "T" through "tor" (i.e., The Text Editor). *CH* verifies the deletion in line 13. In line 14, the user inserts "It" between the two blanks which precede "is". *CH* verifies the insertion in line 15. In line 16, the user asks *CH* to locate "output". *CH* displays the appropriate window in line 17. In line 18, the user inserts !s before !i in order to obtain a blank line between paragraphs.

```
19. tput.!s!iIf you should ha 326
20. f
21. ve enquiries, please cont 351
22. f
23. act I.P. Sharp Associates 376
24.     mLimited
25. to follow -
26. representative.
27. tact  or your local SHARP 375
28.     ////
```

In line 19, *CH* verifies the insertion made in line 18. In line 20, the user asks to move forward one window (25 characters). *CH* displays the next window in line 21. The user asks to move forward again (line 22). *CH* moves forward another 25 characters and displays the appropriate window in line 23. In line 24, the user requests that the characters "I" through "Limited" be moved (to appear as the first line of the address block). *CH* prompts for position of the move in line 25. In line 26, the user indicates that the characters to be moved are to follow "representative". *CH* verifies the move by displaying the revised window. In line 28, the user deletes two blanks and the characters "or".

```
29.  tact your local SHARP APL 375
30.                  /////////!msharp apl!n
31.  ocal !msharp apl!n repres 386
32.  f
33.  entative.I.P. Sharp Assoc 411
34.          / or:!hl5!s
35.  ative or:!hl5!sI.P. Sharp 414
36.  f
```

In line 29, *CH* verifies the deletion requested in line 28. In line 30, the user replaces SHARP APL with !msharp apl!n. *CH* verifies the replacement in line 31. In line 32, the user asks to move forward one window. *CH* displays the next window in line 33. (Here the user runs into the characters which were moved previously.) In line 34, the user deletes the "." and replaces it with a space and the characters "or:!h15!s". This will start the hanging indent for the address block. *CH* verifies the replacement in line 35. In line 36, the user asks to move forward one window.

```
37.   Associates Limited!s!s!t 439
38.                       i
39.  433
40.  !sl45 King Street West!sSuite 1400!sToronto,
41.  478
42.  Canada!sM5H 1J8!s(416) 364-5361!h00
43.  513
44.
```

In line 37, *CH* displays the next window of text as requested in line 36. In line 38, the user asks to enter insertion mode before the !s in order to enter the remainder of the address block. In line 39, *CH* indicates that the text to be entered will follow character 433 in *RINP*. In line 40, the user begins entering text and hits CR. In line 41, *CH* gives the number of the last character entered and waits for more input. The user enters more text (line 42) and hits CR. *CH* prompts for more text in line 43. The user hits CR again (line 44) to exit from insertion mode.

```
45.  Sharp Associates Limited! 434
46.  f
47.  sl45 King Street West!sSu 459
48.  f
49.  ite l400!sToronto, Canada 484
50.  f
51.  !sM5H lJ8!s(416) 364-5361 509
52.  f
53.  !h00!s!s!t7Yours sincerel 534
54.      i
55.  513
56.  !s!iThank you for your interest in !msharp apl!n.
57.  562
58.
59.  -5361!h00!s!iThank you fo 529
60.
```

In line 45, *CH* displays the window where insertion began. In lines 46-53, the user asks *CH* to move forward one window at a time, checking to see that input is correct. In line 54, the user asks to enter input mode after !h00 in order to enter the last paragraph. *CH* prompts for input in line 55. The user enters the text for the last paragraph in line 56. *CH* prompts for more input in line 57. The user hits CR (empty input line) to exit from input mode in line 58. In line 59, *CH* displays window where input began line 59. At this point, the user decides that all necessary modifications have been made. He then enters a blank line (hits CR) to exit from *CH* in line 60.

On inspection of the previous terminal session, the user notes that there is still one error which needs correcting. A !s was omitted after the !h15. He then re-enters *CH*, locates the correct region, and inserts the missing !s using the following sequence:

```
      CH
!t7November 4, 1977.!sAcm 25
lative or:
ative or:!hl5!sI.P. Sharp 414
             .!s
:!hl5!s!sI.P. Sharp Assoc 422
```

RINP should now contain the raw text needed to produce the second letter. After exiting from *CH* by entering a blank line, the user proceeds once again to stage 2. Upon entering the function *EDIT*, the formatted letter, which appears at the beginning of this section, will be printed. Note that two passes will be required since two typefaces have been requested. This will be indicated by the blanks left in the letter when it is first printed and by the fact that the print element stops in the middle of the page after the first pass.

It is important to understand the difference between text correcting instructions and edit control commands. Text correcting instructions correspond to the first non-blank character which is entered on the line below any window of text displayed by *CH*. The text correcting instructions:

l	locate
.	insert
d	delete
f	forward
i	insert mode
m	move
/	delete

used in the sample terminal session above are not (and never become) part of the raw text in *RINP*. They are used only to move through it and to modify it. By contrast, edit control commands **are** a part of the raw text. They are entered along with the text of the document during the input stage. When *EDIT* is executed, the edit control commands embedded within the text are recognized and removed. They instruct the Editor on how to format the text, but do not appear in the formatted output. A complete list of edit control commands and an explanation of their use can be found in Appendix I. Text correcting instructions are summarized in Appendix II.

THE STATE-SETTING FUNCTIONS

The previous sections of this manual have demonstrated how the *EDIT* function uses the edit control commands contained in the raw text to produce formatted output in stage 2. In addition, *EDIT* relies on a set of "state setting functions" to format and print a document. When the Text Editor is first loaded, these state setting functions are in a default condition and remain in this condition until the user specifically requests that one or more of them be changed. Up until now these functions have always been in a default condition when the *EDIT* function was executed. To ascertain the current condition of the state setting functions, the user enters *STATE*. For example, entering *STATE* in a default condition will produce the following:

```
      STATE
SOURCE IS RINP
PRINT
NORMAL
WIDTH  60
DEPTH  55
RAGGED
OUTFILE 0
WINDOW 25
PRINCIPAL 'COURIER'
```

Any one of these eight conditions can be changed by entering an appropriate alternative. Each of the above default conditions and their alternatives are described below.

1. *SOURCE*: The default here is *RINP*. This indicates that the raw text to be edited is stored in *RINP*. Alternatively, raw text can be stored in a file on the SHARP APL File Subsystem. In this case, the file containing the raw text is specified as follows:

   SOURCE I J K where *I* is the tie number of the file in which the raw text is stored. *J* is the first component containing data and *K* is **one more** than the last component containing data.

   To reset *SOURCE* to its default condition, enter *SOURCE* 0. The use of *SOURCE* to specify a file for the raw text is explained in more detail in the next section of this manual.

2. *PRINT*: The alternative to *PRINT* is *NOPRINT*. This function determines whether or not the formatted output produced with *EDIT* will be printed on the terminal. Enter *PRINT* or *NOPRINT* to change conditions.

3. *NORMAL*: The alternative to *NORMAL* is *TRANSLIT*. *TRANSLIT* enables the printing of edited output using a print element of a different terminal transmission code from the type normally used with a particular terminal. If, for example, the output from *EDIT* does not correspond to the given input, *TRANSLIT* should be employed. To reset the default condition, enter *NORMAL*.

4. *WIDTH*: The default of *WIDTH* is 60 and the alternative can be any number n from 15 to 130. *WIDTH* determines the maximum number of characters which will appear on a line. To change conditions, enter:

   WIDTH n

   Note that the APL system command)*WIDTH* has to have been set to a value at least equal to that set with *WIDTH*.

5. *DEPTH*: The default of *DEPTH* is 55. *DEPTH* specifies the number of lines which will be printed on a page before a break occurs. To change the value, enter

```
DEPTH n
```

where n is the number of lines to appear on each page.

6. *RAGGED*: The alternative to *RAGGED* is *SMOOTH*. If *SMOOTH* is entered, each line of text will be expanded so that the last character occurs at the right margin (i.e., the text is right justified). No more characters will appear on any given line than is the case with *RAGGED*; rather the number of spaces between words will be increased to accomplish the smoothing.

7. *OUTFILE*: The default, *OUTFILE* 0, indicates that the formatted output produced with the function *EDIT* is not to be filed in the SHARP APL File Subsystem. To have the formatted output filed, enter

```
OUTFILE I
```

where *I* is the tie number of the file in which the formatted output is to be stored. Note that the combination of *NOPRINT* and *OUTFILE* 0 is a futile combination since the output produced with *EDIT* will neither be printed nor filed. This function will be explained in more detail in the next section of this manual.

8. *WINDOW*: The default 25 refers to the number of characters shown in each *CH* window during text modification. To change this condition, enter

```
WINDOW n
```

9. *PRINCIPAL*: The default condition is *COURIER* and the alternative is *APL*. This function is used to specify the principal typeface to be used and to ensure that underlining is carried out correctly in the primary font and that certain other overstrikes (i.e., exclamation mark) will be printed properly. To change from the default, enter *PRINCIPAL* *'APL'*; to restore it enter *PRINCIPAL* *'COURIER'*.

Changing two of these default conditions will demonstrate how the state setting functions can alter the output produced with *EDIT*. For example, if the user wishes the sample letter from the previous section to be both left and right justified and have a width of 50 characters per line, he enters

```
SMOOTH
WIDTH 50
```

Then, when he executes *EDIT*, the following output is produced.

November 4,
1977.
Acme Products Limited,
123 Main Street,
Waxen Apple, Saskatchewan.

Dear Gentlemen:

This is to introduce you to the Text Editor of the *SHARP APL* system, available from I.P. Sharp Associates Limited. It is a program package designed to help the user prepare edited output.

If you should have enquiries, please contact your local *SHARP APL* representative or:

I.P. Sharp Associates Limited
145 King Street West
Suite 1400
Toronto, Canada
M5H 1J8
(416) 364-5361

Thank you for your interest in *SHARP APL*.

Yours
sincerely,

Ralph Morrison.

The output looks peculiar because "1977" and the word "sincerely" have gone to new lines. The reason for this is that the tab value specified with !t7 is too great for the shorter length lines (*WIDTH* 50). Therefore, the user re-enters *CH* and replaces all occurrences of !t7 with !t6.

```
      CH
!t7November 4, 1977.!sAcm 25
r!t7
replace by -
!t6
!t6November 4, 1977.!sAcm 25
!t6Yours sincerely,!s!s!s 543
!t6Ralph Morrison.        673
```

The text correcting instruction "r" is used to request that all subsequent occurrences of !t7 be replaced with !t6. Note that *CH* verifies each replacement by displaying the appropriate windows. Now when *EDIT* is executed, the new letter will be printed as follows:

November 4, 1977.

Acme Products Limited,
123 Main Street,
Waxen Apple, Saskatchewan.

Dear Gentlemen:

This is to introduce you to the Text Editor of the *SHARP APL* system, available from I.P. Sharp Associates Limited. It is a program package designed to help the user prepare edited output.

If you should have enquiries, please contact your local *SHARP APL* representative or:

I.P. Sharp Associates Limited
145 King Street West
Suite 1400
Toronto, Canada
M5H 1J8
(416) 364-5361

Thank you for your interest in *SHARP APL*.

Yours sincerely,

Ralph Morrison.

USING THE SHARP APL FILE SUBSYSTEM IN EDITING

The state setting functions described above have indicated the ways in which the SHARP APL File Subsystem can be used in the editing process. Using the file handling functions available in 4 *EDIT*, both the raw text input in stage 1 and the formatted output produced in stage 2 can be stored on file. In this way, large quantities of raw text can be stored and then accessed whenever modifications are necessary. In addition, the file on which edited output from stage 2 is stored, can be used to produce subsequent copies of the document without the cost of editing it each time.

In order to demonstrate the use of the file handling functions in the editing process, the raw text for the document which appears below and on the following page will be entered using the Text Editor and stored in a file. The formatted output produced in stage 2 will be stored in a second file. The file operations used in this section of the manual are described in detail in the I.P. Sharp manual entitled "SHARP APL File Subsystem Manual".

```
      1977-78 CONCERT SERIES: THIRD CONCERT

January 28, 1978                              8:30 P.M.


The Committee for Cultural Consciousness thanks you for the
support you have shown by coming here tonight.

We hope that the four selections which will be performed
this evening will provide those unfamiliar with the music of
the Baroque masters with both an educational and pleasurable
cultural experience.
```

PROGRAM FOR THIS EVENING

CONCERTO FOR VIOLIN AND STRINGS NO. 2 IN E MAJOR — BACH (1685-1750)

Allegro
Adagio
Allegro assai

CONCERTO GROSSO IN D MINOR OP. 3, NO. 11 — VIVALDI (1678-1741)

Allegro
Largo e spiccato
Allegro

CONCERTO FOR OBOE AND STRINGS OP. 9, NO. 2 IN D MINOR — ALBINONI (1671-1750)

Allegro non presto
Adagio
Allegro

CONCERTO FOR FLUTE AND STRINGS IN G MAJOR — QUANTZ (1697-1773)

Allegro
Arioso
Presto

Before the user enters stage 1 and begins to enter this document, he must first create two files; one to store the raw text and one to store the output produced with *EDIT*. To create an empty file called *SOURCE* for the raw text, the user enters

```
'SOURCE' ⎕CREATE 1
```

Then, to create an empty file called *OUTPUT* on which to store the edited output, he enters

```
'OUTPUT' ⎕CREATE 2
```

Once the appropriate files have been created, the user loads ⍧ *EDIT*, executes *CH* and begins entering the raw text for the document. The terminal session would look as follows:

```
      'SOURCE' ⎕CREATE 1
      'OUTPUT' ⎕CREATE 2
      )LOAD ⍧ EDIT
      CH
0
!hl0!wl0!s!ml977_78 concert series. third concert!n!j!h00!w00
61
!s!sJanuary 28, 1978!18:30 P.M.!r!s!sThe Committee for Cultural
125
Consciousness thanks you for the support you have shown by
184
coming here tonight.!s!sWe hope that the four selections
241
which will be performed this evening will provide those
297
unfamiliar with the music of the Baroque masters with both an
359
educational and pleasurable cultural experience.!s!p!mprogram
421
for this evening!n!c!s!sCONCERTO FOR VIOLIN AND STRINGS!l
479
BACH!rNO. 2 IN E MAJOR!l(1685-1750)!r!h04!sAllegro!s
531
Adagio!sAllegro assai!h00!s!s!sCONCERTO GROSSO IN D MINOR!l
590
VIVALDI!rOP. 3, NO. 11!l(1678-1741)!s!h04!sAllegro!sLargo
647
e spiccato!sAllegro!h00!s!s!sCONCERTO FOR OBOE AND STRINGS!l
708
ALBINONI!rOP. 9, NO. 2 IN D MINOR!l(1671-1750)!s!h04!sAllegro
769
non presto!sAdagio!sAllegro!h00!s!s!sCONCERTO FOR FLUTE AND
829
STRINGS!lQUANTZ!rIN G MAJOR!l(1697-1773)!r!h04!sAllegro!s
887
Arioso!sPresto!h00
905
```

Several new edit control commands have been introduced in the above session:

!w is the right hanging indent which corresponds to the left hanging indent !h. It is cancelled with !w00.

!j expands the string which **precedes** it so that it fills the available space.

!c centres the string which **precedes** it.

!l left justifies the string which **precedes** it.

!r right justifies the string which **precedes** it. The string which follows automatically starts on the next line.

!p forces a new page starting with the string which **precedes** it. Therefore, !s!p is used to indicate a page break.

In the terminal session above, !h10 and !w10 are used to set up new margins of 10 spaces on either side of the page. The heading which appears in the APL typeface is justified within these margins using !j. !h00 and !w00 terminate the margins set for the heading. !l and !r are used to left and right justify respectively the text which precedes them. The text for the first page is entered followed by !s!p which forces a new page.The title which appears on the second page is centred with !c.

The edit control commands, !xl, !xr and !xc, which do not appear above, are used to request multiple left justified, multiple right justified and multiple centred lines, respectively. !xl is the default condition and terminates either !xr or !xc.

Once all the raw text for the document has been entered into *RINP*, the user is ready to have it stored in the file called *SOURCE* using the file handling function *PUT*. When he enters

```
PUT 1
```

the Editor stores the contents of *RINP* on the file **tied to 1** and responds with the message

```
NEW COMPONENT NO. 1
```

indicating that the raw text has been stored in component number 1 of the file. The file handling function *PUT* transfers the contents of *RINP* to the next available component of the specified file (in this case, the file tied to 1).

When there is a large amount of raw text to enter, it may be necessary to enter *PUT* several times. Each time this function is executed, the contents of *RINP* are stored in the next available component of the specified file and the variable *RINP* is automatically cleared so that more text can be entered.

Once all the raw text has been stored on file, the user can proceed to stage 2 and have it edited. However, before executing *EDIT*, it is necessary to tell the Editor that the raw text is now in the File Subsystem. To do this, the user resets the *SOURCE* parameter of the *STATE* by entering

```
SOURCE 1 1 2
```

This tells the Editor that the raw text can be found in the first component of the file tied to 1. The first number is the tie number of the file. The second number is the number of the first component holding raw text. The third number is the number of the **last component plus 1** holding raw text.

At this point, the user also wishes to indicate that the edited document should be stored in the file called *OUTPUT* which is tied to 2. To do this, he resets the *OUTFILE* parameter of the *STATE* by entering

```
      OUTFILE 2
```

The Editor responds with a message indicating the present number of components in the file

```
EDITED TEXT TO FILE '1234567 OUTPUT'; PRESENT LIMITS ARE  1 1
```

(Limits of 1 1 mean that the file is currently empty.)

Now that the *STATE* has been reset to indicate the location of the raw text and to specify the file on which the output is to be stored, the user enters the function *EDIT*. The formatted document will be printed on the terminal and, at the same time, stored on the file called *OUTPUT*.

To verify that the file tied to 2 now contains the edited document, the user enters

```
      □SIZE 2
```

The Editor responds with the number of components now on the file

```
    1  5  4530  51340
```

The numbers 1 5 indicate that 4 components of the file tied to 2 have been used. Note that the second number (5) is one greater than the last component number. There are two components for each page of output: one is for the first typeface, the second is for the alternate typeface. Remember that in the sample document, both pages contained a small amount of text which was to be printed in the alternate typeface. A look into file 2 will reveal that components 2 and 4 contain character vectors composed primarily of carriage returns. Components 1 and 3 hold character vectors containing most of the text for the two pages.

If the user wishes to print his two page document again, he can employ another of the file handling functions in 4 *EDIT*. The function *RUNOUT* prints text that was previously edited and stored in the file specified with *OUTFILE*. In this case, a copy of the formatted document is now stored in the first 4 components of the file tied to 2. When the user enters

```
      RUNOUT 2 1 5
```

the two page document stored in components 1, 2, 3, and 4 of the file tied to 2 will be printed. As with the function *SOURCE*, the third element of the right argument vector of *RUNOUT* is one greater than the last component number.

Suppose that, at this point, the user wishes to alter his document and add the line "** Soloists: Mildred Tupple, flute; D. Kerona, oboe." to the bottom of page 2. The following sequence would be used to retrieve the raw text, alter it, replace the old raw text in the File Subsystem, destroy the previous contents of the outfile, create and file the new formatted output and then print only the page which has been modified (page 2).

```
      GET 1 1
CUR = 1 1
      CH
!hl0!wl0!s!ml977_78 conce 25
z
905
!s!s!s!s!s!s**Soloists: Mildred Tupple, flute; D. Kerona, oboe.
969
CR
o!h00!s!s!s!s!s!s**Soloi 925
CR
```

```
      PUT 1 1
TEXT REPLACED
      ⎕DROP 2 ¯4
      ⎕SIZE 2
1  1  0  51340
      STATE
SOURCE IS FILE TIE NO. 1, COMPONENTS 1 THROUGH 1
PRINT
NORMAL
WIDTH 60
DEPTH 55
RAGGED
OUTFILE 2
WINDOW 25
PRINCIPAL 'COURIER'
      NOPRINT
      EDIT
      RUNOUT 2 3 5
```

The command *GET 1 1* tells the Editor to get the first component of the file tied to 1 and put it back into the variable *RINP*. The Editor indicates that this has been accomplished with the message *CUR = 1 1*. The user then uses *CH* to alter the contents of *RINP*. Note that the text correcting instruction z has been introduced here. z is used to enter input mode when the new material to be input is to appear at the **end** of the existing document. Once the contents of *RINP* have been altered, the user enters the command *PUT 1 1* which tells the Editor to **replace** the first component of the file tied to 1 with the revised raw text in *RINP*. The Editor indicates that the component has been replaced with the message *TEXT REPLACED*.

Note: Both the file tie number and the component number must be specified when using *PUT* to **replace** a component of raw text. Entering *PUT 1* would put the altered contents of *RINP* into the next available component (in this case, the second component) of the source file.

The old edited version of the document is dropped from the file tied to 2 with ⎕*DROP 2 ¯4*. This eliminates the **last** four components of the file and restores it to the initial empty condition. To verify that the outfile is now empty, the user enters ⎕*SIZE 2* and the new limits are printed. He then enters *STATE* to ascertain that the settings are correct. *PRINT* is reset to *NOPRINT* since the user does not want the document to be printed on the terminal when *EDIT* is executed. The function *EDIT* is then executed and a formatted copy of the revised document is stored on the output file. Finally, the modified second page (components 3 and 4 of the file) is printed with *RUNOUT 2 3 5* as follows:

PROGRAM FOR THIS EVENING

CONCERTO FOR VIOLIN AND STRINGS — BACH
NO. 2 IN E MAJOR — (1685-1750)

Allegro
Adagio
Allegro assai

CONCERTO GROSSO IN D MINOR — VIVALDI
OP. 3, NO. 11 — (1678-1741)

Allegro
Largo e spiccato
Allegro

CONCERTO FOR OBOE AND STRINGS — ALBINONI
OP. 9, NO. 2 IN D MINOR — (1671-1750)

Allegro non presto
Adagio
Allegro

CONCERTO FOR FLUTE AND STRINGS — QUANTZ
IN G MAJOR — (1697-1773)

Allegro
Arioso
Presto

**Solists: Mildred Tupple, flute; D. Kerona, oboe.

APPENDIX I

Summary of Edit Control Commands

Form		Description
APL	**Courier**	
¨	!	¨ or ! followed by a blank is used to achieve the editing character ¨ or ! in the text. ¨ and ! are achieved by hitting upper case 1 on the terminal. On some terminals, the ! may be printed as plus over minus.
¨*C*	!c	Centre **preceding** string on the page. This command is often used after a heading.
¨*HNN*	!h*NN*	Start hanging indentation of *NN* spaces to the **left**. Terminated by ¨*H*00 or !h00.
¨*I*	!i	Start a new paragraph by forcing a new line and indenting it 5 spaces.
¨*J*	!j	Fill line with **preceding** string.
¨*L*	!l	Left justify the **preceding** string. This is used on the same line as ¨*R* or !r to separate the text into units which will be left and right justified as specified.
¨*M*	!m	Change to alternate typeface. Used in order to permit two fonts. Terminated by ¨*N* or !n on the same page.
¨*N*	!n	Return to normal mode by negating whichever occurred most recently of either ¨*M* or ¨*U* (!m or !u).
¨*P*	!p	Start a new page. As this takes **preceding** line as beginning of new page, ¨*S* ¨*P* (!s!p) is used to start a new page.
¨*QNN*	!q*NN*	End the page if fewer than *NN* lines available on it. Used to avoid spillage of a paragraph to next page.
¨*R*	!r	Right justify **preceding** string. Forces a new line for following string.
¨*S*	!s	Start a new line.

¨*TN*	!t*N*	Tabulate to tab number *N*+1. This edit control command employs the value given by global variable *TABS*[*N*+1]. The default condition is *TABS*←0 , 5×⍳9. When the variable *TABS* is set to other than its default condition, the APL command)*TABS N* should be used to set the system tabs where *N* corresponds to the difference between consecutive elements of the vector *TABS*.
¨*U*	!u	Start underlining text which follows. Terminated by ¨*N* or !n.
¨*WNN*	!w*NN*	Start hanging indentation of *NN* spaces on the **right**. Terminated by ¨*W*00 or !w00.
¨*XC*	!xc	Multiple centred lines starting with **preceding** text. Terminated by ¨*XL* or !xl.
¨*XL*	!xl	Multiple left justified lines starting with **preceding** text. Terminates ¨*XR* ¨*XC* or !xr !xc. ¨*XL* !xl is the default condition.
¨*XR*	!xr	Multiple right justified lines starting with **preceding** text. Terminated by ¨*XL* or !xl.
⍢	G	Used as a character to designate a forced blank space. In Courier it is an upper case G overstruck with uppercase T.

APPENDIX II

Summary of Text Correcting Instructions

Type 1: Window Moving Instructions

These instructions may be in upper or lower case.

BNNN	Move window back *NNN* characters.
B	Move back one window.
FNNN	Move window forward *NNN* characters.
F	Move forward one window.
NNN	Move window to start at character *NNN*.
LSTRING	Locate next occurrence of *STRING*; move window to it.

Type 2: Window Dependent Instructions

//STRING	Replace characters above one or more slashes by *STRING*.
\\STRING	Identical to //.
.STRING	Insert *STRING* before character above the period .
DSTRING	Delete text from the character above the *D* through the next occurrence of *STRING*.
I	Characters are to be inserted before the character above *I*. A return to input mode of function *CH* is effected and until this is terminated by an empty line (two consecutive carriage returns) all characters entered are inserted in the position designated.
MSTRING	Moves all text from the character above the *M* through the next occurrence of *STRING* to the right. The exact location is specified following prompt issued by *CH*.
PSTRING	Print all characters starting from that above P through *STRING*.
P	Print all characters from that above P to end of text.

Type 3: Window Independent Instructions

RSTRING — Replaces all subsequent occurrences of the sub-string *STRING* with a new sub-string which is specified following a prompt from *CH*.

Z — Enter input mode of function *CH*. Like *I*, except that the new text goes at the **end** of present text.

Note that once an instruction has been entered under a character in the window, it is not possible to enter another text correcting instruction under another character to the right, as the new instruction will be construed as being part of the first.

For simple corrections to text (such as mis-spelled words, erroneous punctuation or capitalization, etc.) the *R* instruction is probably the most useful.

Note that the sub-string *STRING* is used in the above instructions in two very distinct ways. In the / and . instructions, it is a sub-string of text to be inserted; in the *D*, *L*, *M*, *P* and *R* instructions, it is a location finding string.

APPENDIX III

Summary of State Setting Functions

Default	Alternative
SOURCE IS RINP (*SOURCE* 0)	*SOURCE I J K*
PRINT	*NOPRINT*
NORMAL	*TRANSLIT*
RAGGED	*SMOOTH*
WIDTH 60	*WIDTH N*
DEPTH 55	*DEPTH N*
OUTFILE 0	*OUTFILE I*
WINDOW 25	*WINDOW N*
PRINCIPAL '*COURIER*'	*PRINCIPAL* '*APL*'

APPENDIX IV

Summary of File Handling Functions

1) *GET I J* - Places a copy of the raw text stored in component *J* of the file tied to *I* in the variable *RINP*. The raw text can then be modified using *CH* or edited using *EDIT*.

2) *PUT I* - Stores the raw text contained in the variable *RINP* in the first available component of the file tied to *I*. *RINP* is automatically cleared of its contents.

3) *PUT I J* - Replaces the contents of component *J* of file *I* with the contents of *RINP*. If the first 50 characters of *RINP* are not identical to the first 50 characters stored in the component, a second request for replacement will be prompted for before the transfer takes place. If the component number specified with *J* does not correspond to the number of a component holding data, then the raw text will be appended to the file.

4) *RUNOUT*: Permits the printing of text which was previously edited and stored in the file specified with *OUTFILE*. The syntax of this function is:

```
RUNOUT I J K
```

where *I* is the tie number, *J* is the first component number, and *K* is one more than the last component number of the file containing edited output. The contents of the file components are printed according to the convention of the Text Editor: two components for each page of output - one for each typeface. If the second component is empty, the type element will come to rest at the left margin. Otherwise, it will stop in the centre of the page. An entire page may be reprinted by entering the character " (or !) when the keyboard unlocks between pages.